CS 639: Foundation Models
Deep Learning II

Fred Sala

University of Wisconsin-Madison

Jan. 29, 2026

Announcements

- Midterm: **Weds. March 11th**
  - HW 1: Coming out **next Thursday**
- **Resources**
  - https://www.deeplearningbook.org/ : Solid intro to DL
- Class roadmap:

Thursday Jan. 20	Deep Learning II
Tuesday Feb. 3	Self-Supervised Learning
Thursday Feb. 5	Guest Lecture
Tuesday Feb. 10	Transformers and Attention I

Start FMs and Arch

Outline

- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures (mostly from last time)
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks
- **Graph Models**
  - Data relationships, graph neural networks, graph convolutions

Outline

- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures (mostly from last time)
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks
- **Graph Models**
  - Data relationships, graph neural networks, graph convolutions

How to classify

Cats vs. dogs?

Dual

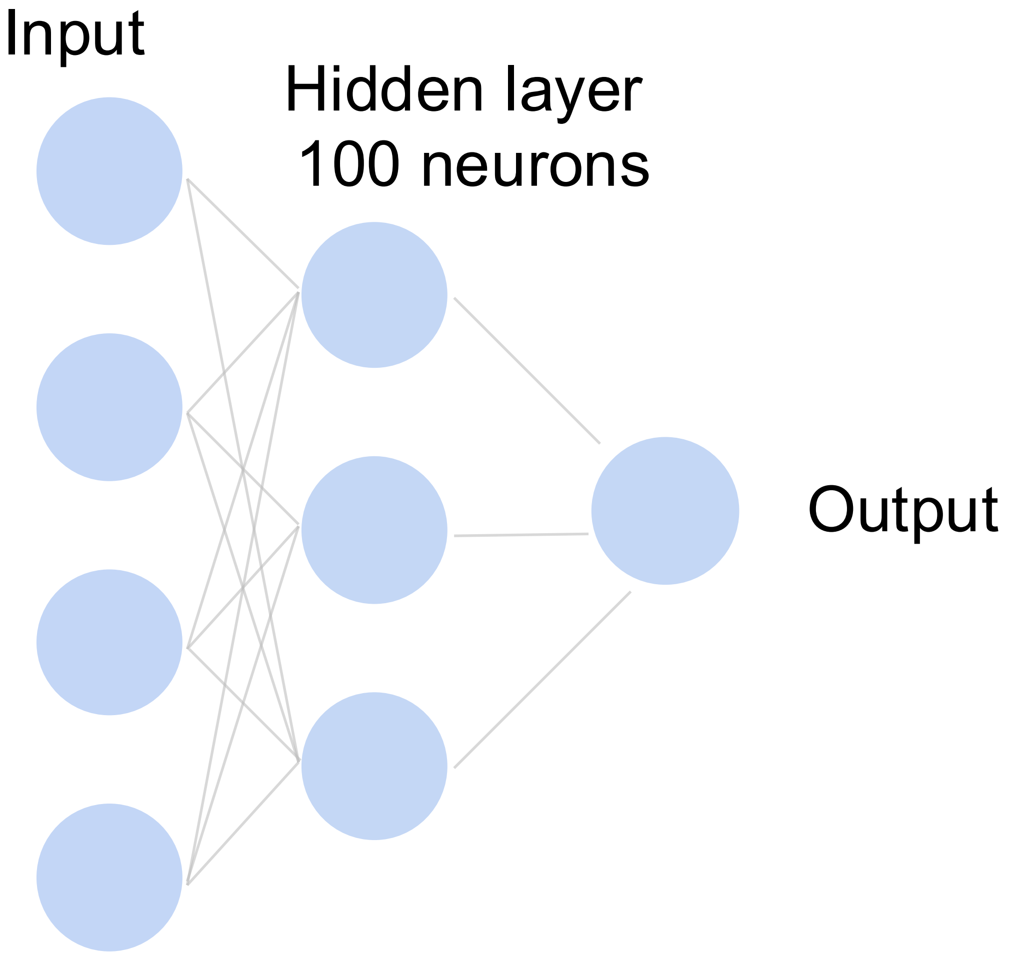
12MP

wide-angle and
telephoto cameras

36M floats in a RGB image!

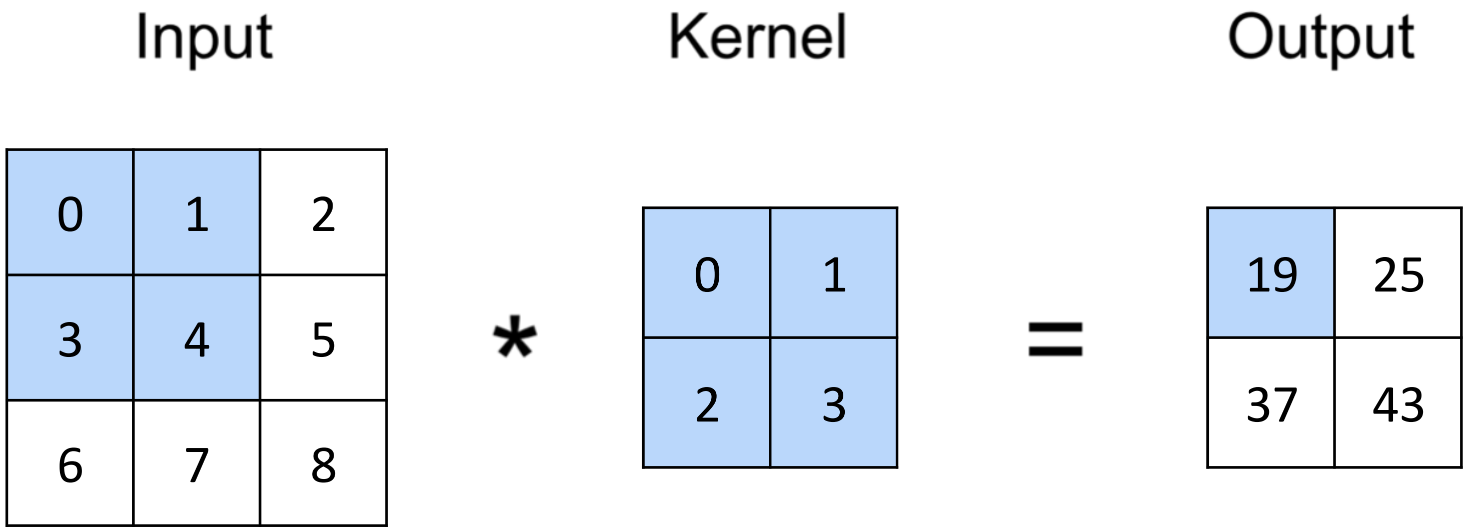
Fully Connected Networks (From Last Time)

Cats vs. dogs?



~ 36M elements x 100 = ~**3.6B** parameters!

2-D Convolution



$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

2-D Convolution

Input

0	1	2
3	4	5
6	7	8

*

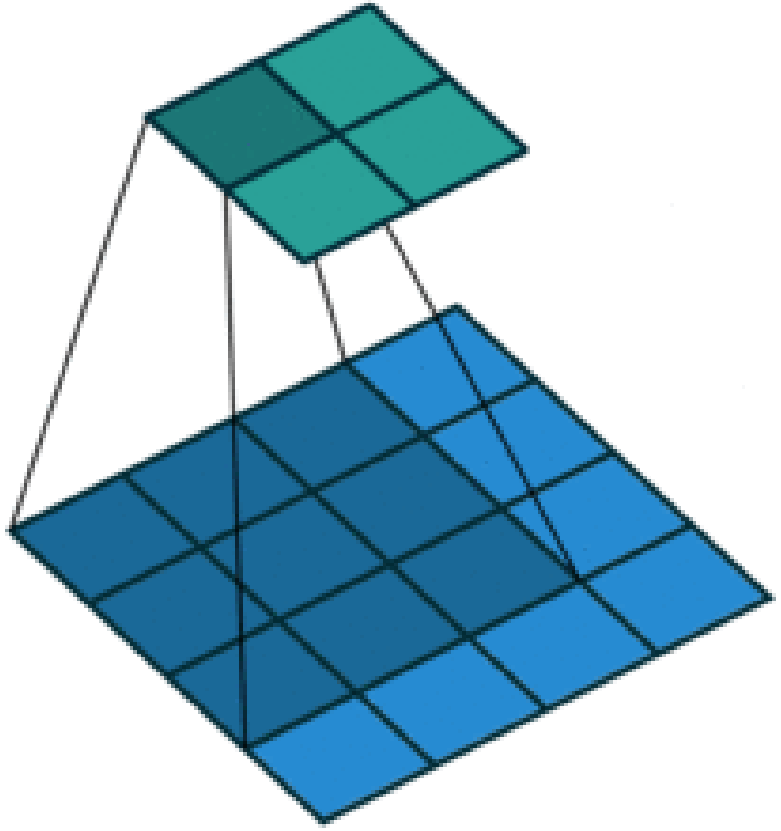
Kernel

0	1
2	3

=

Output

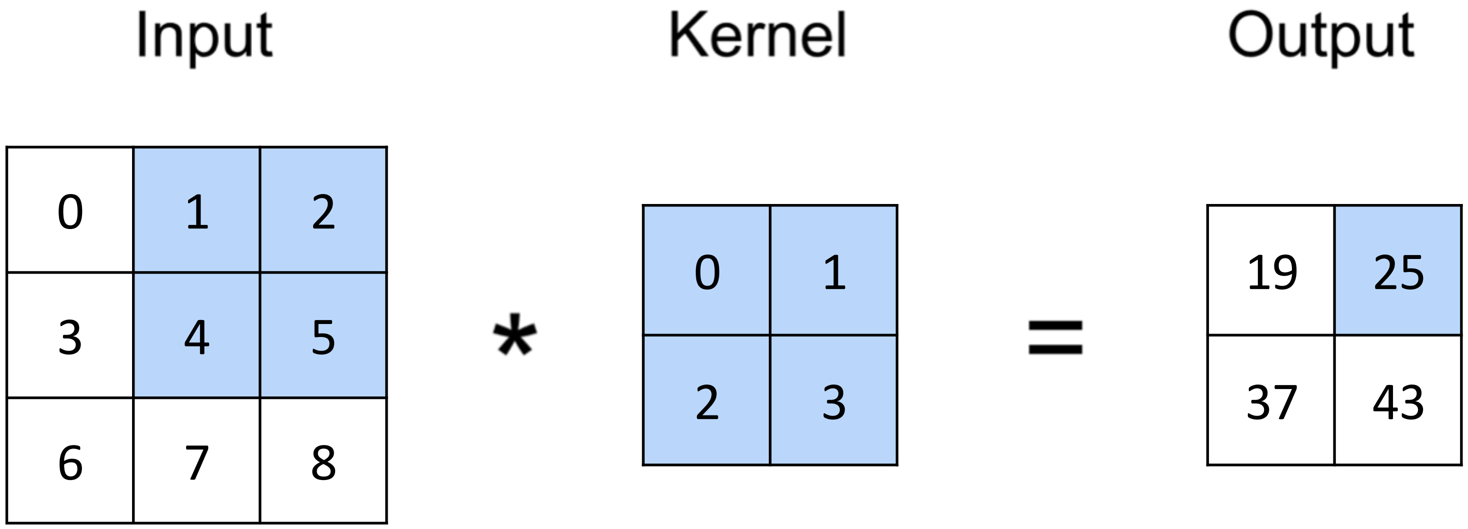
19	25
37	43



$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

(vdumoulin@ Github)

2-D Convolution



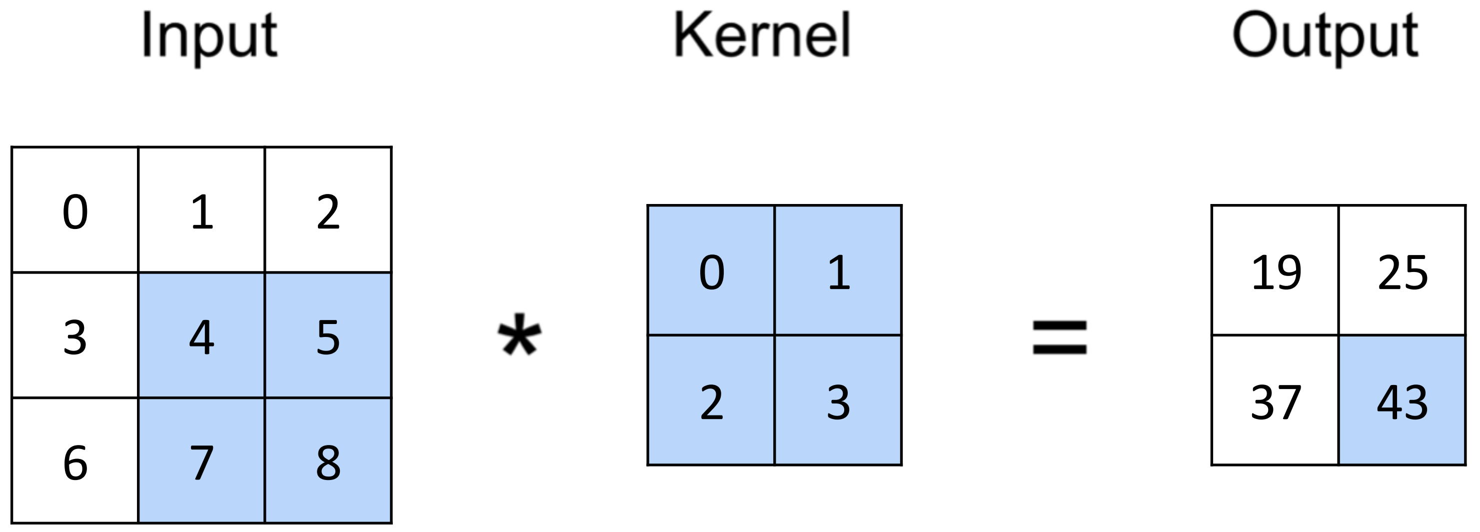
1x0 + 2x1 + 4x2 + 5x3 = 25

2-D Convolution

$$\underset{\text{Input}}{\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}} * \underset{\text{Kernel}}{\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}} = \underset{\text{Output}}{\begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}}$$

$$3x0 + 4x1 + 6x2 + 7x3 = 37$$

2-D Convolution



$$4\text{x}0 + 5\text{x}1 + 7\text{x}2 + 8\text{x}3 = 43$$

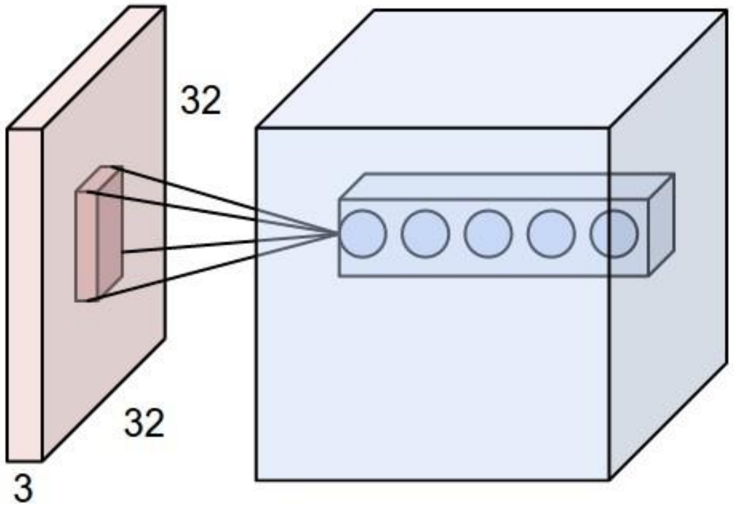
Neural Networks: Convolution Layers

- Notation:
  - *X:* n_h x n_w input matrix
  - *W:* k_h x k_w kernel matrix
  - *b* : bias (a scalar)
  - *Y*: () x () output matrix
- As usual *W*, *b* are learnable parameters

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$$

Neural Networks: Convolution NNs

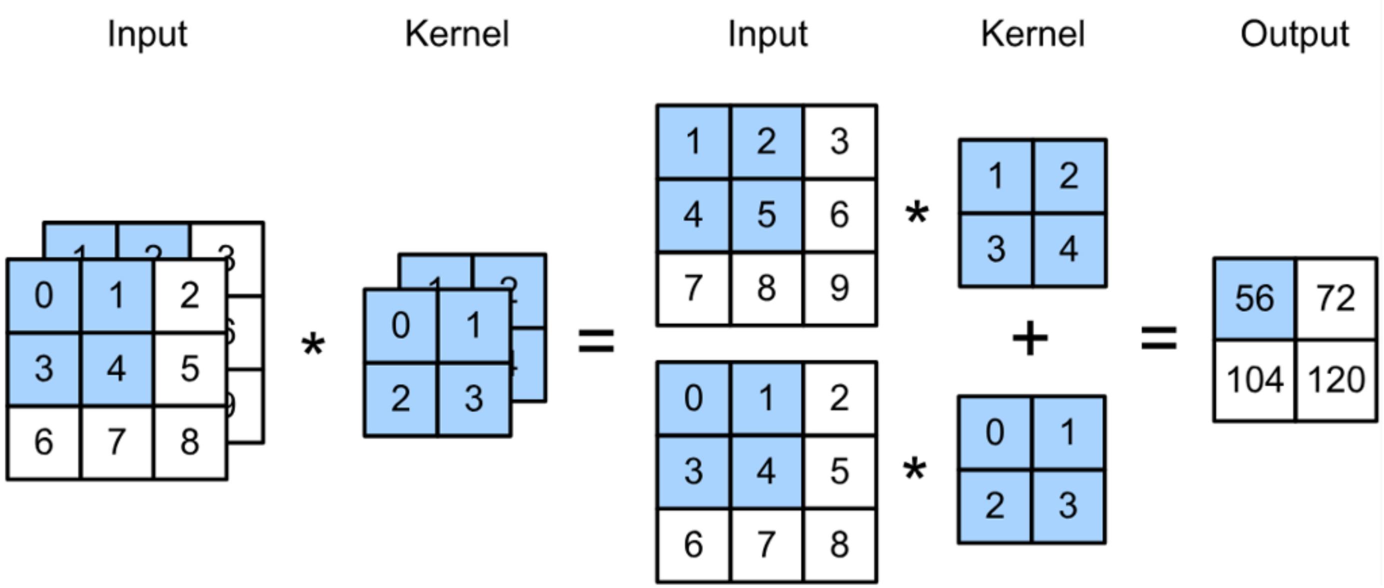
- Properties
  - Input: volume c_i x n_h x n_w (channels x height x width)
  - Hyperparameters: # of kernels/filters c_o, size k_h x k_w, stride s_h x s_w, zero padding p_h x p_w
  - Output: volume c_o x m_h x m_w (channels x height x width)
  - Parameters: k_h x k_w x c_i per filter, total $(k_h$ x k_w x $c_i)$ x c_o



Stanford CS 231n

Multiple **Input** Channels

- Have a kernel matrix for each channel, and then sum results over channels



$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4)$$
$$+(0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3)$$
$$= 56$$

Convolutional Layers: Channels

- How to integrate multiple channels?
  - Have a kernel for each channel, and then sum results over channels

$$\mathbf{X} : c_i \times n_h \times n_w$$

$$\mathbf{W} : c_i \times k_h \times k_w$$

$$\mathbf{Y} : m_h \times m_w$$

$$\mathbf{Y} = \sum_{i=0}^{c_i} \mathbf{X}_{i,:,:} \star \mathbf{W}_{i,:,:}$$

"Slices" of tensors

Tensor: generalization of matrix to higher dimensions

Multiple **Output** Channels

- No matter how many inputs channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates an output channel

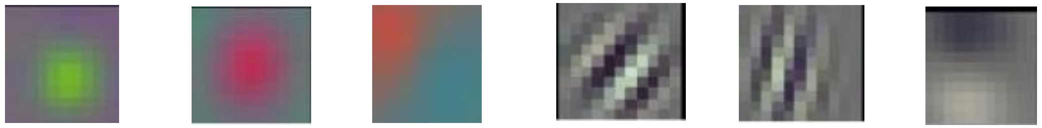
Multiple **Output** Channels

- No matter how many inputs channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates an output channel
- Input $\mathbf{X} : c_i \times n_h \times n_w$
- Kernels $\mathbf{W} : c_o \times c_i \times k_h \times k_w$
- Output $\mathbf{Y} : c_o \times m_h \times m_w$

$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:}$$
$$\text{for } i = 1, \ldots, c_o$$

Multiple Input/Output Channels

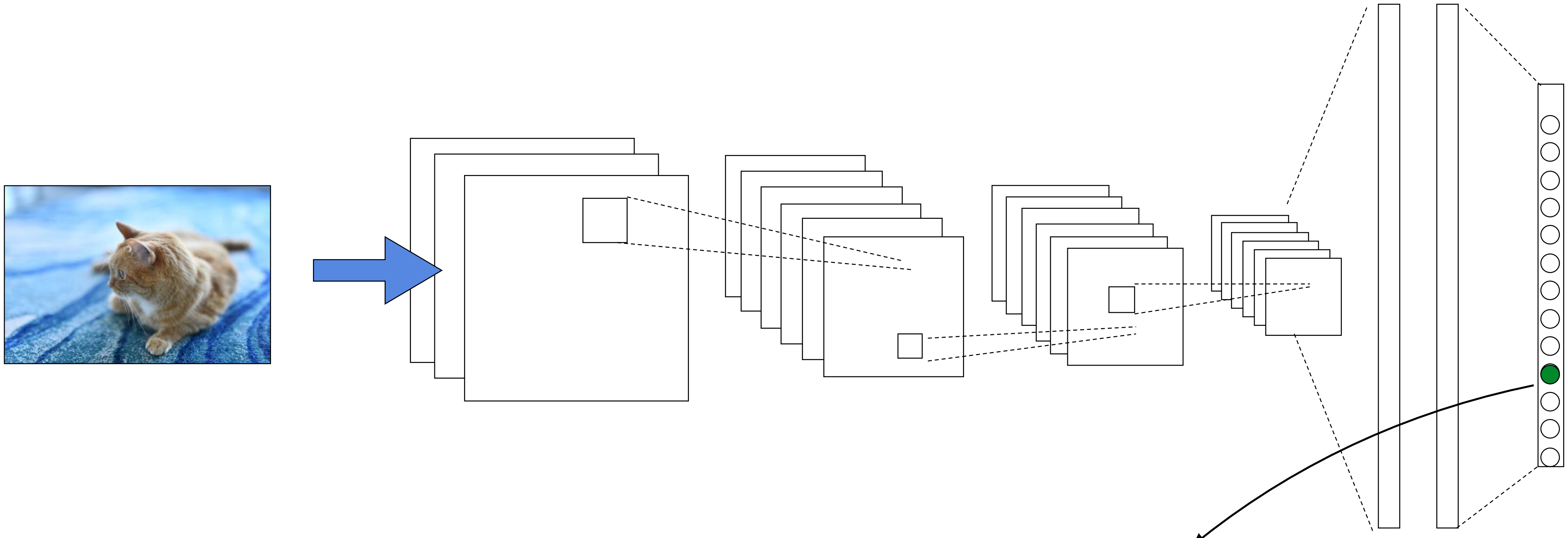
- Each 3-D kernel may recognize a particular pattern



(Gabor
filters)

Training a CNN

- Q: so we have a bunch of layers. How do we train?
- A: same as before. Apply softmax at the end, use backprop.

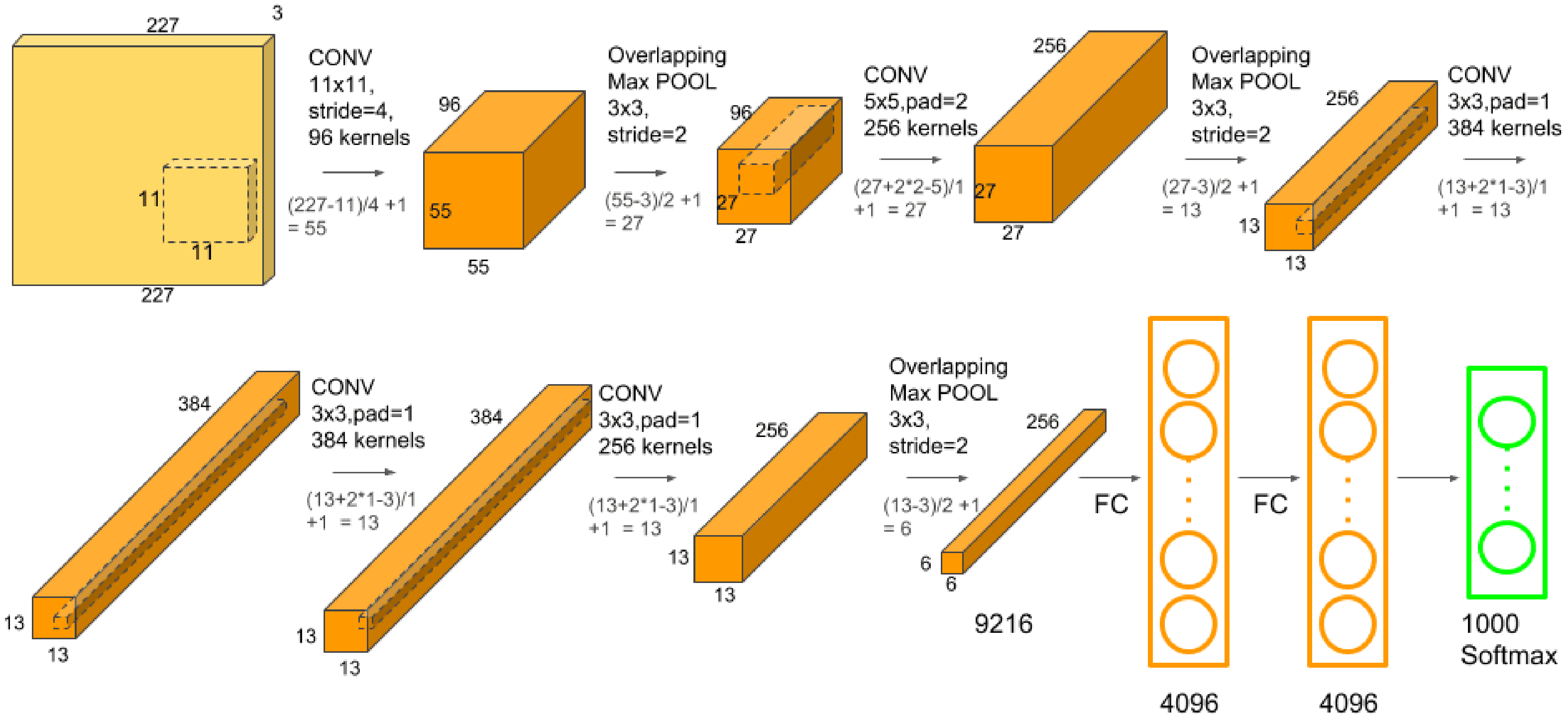


$$p_i(\boldsymbol{x}) = \frac{\exp\left(f_i(\boldsymbol{x})\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})\right)},$$

softmax

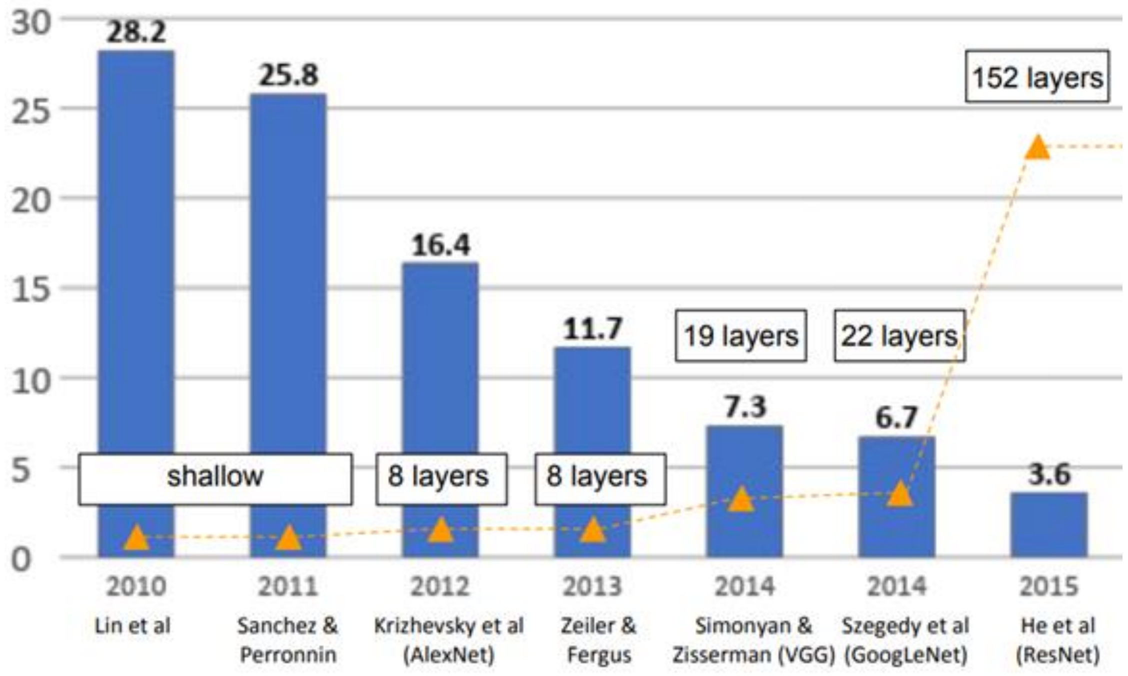
CNN Architectures: AlexNet

- First of the major advancements: AlexNet
- Wins 2012 ImageNet competition
- Major trends: deeper, bigger LeNet



Evolution of CNNs

ImageNet competition (error rate)



Credit: Stanford CS 231n

Data Augmentation

Augmentation: transform + add new samples to dataset

- Transformations: based on domain
- Idea: build **invariances** into the model
  - **Ex**: if all images have same alignment, model learns to use it
- Keep the label the same!

Data Augmentation: Examples

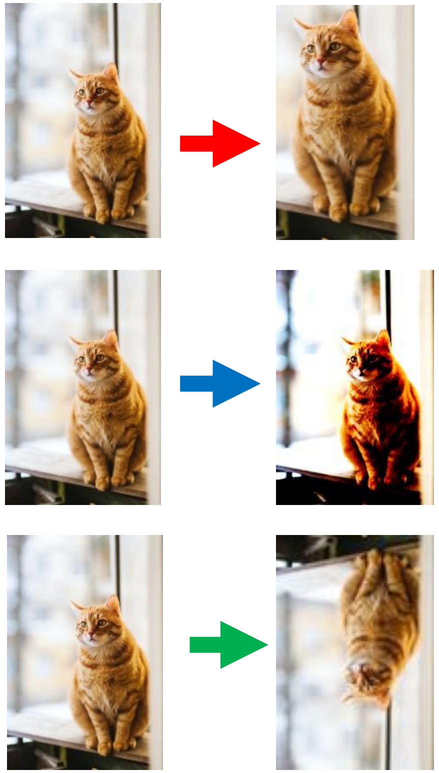
Examples of transformations for images

- **Crop** (and zoom)
- **Color** (change contrast/brightness)
- **Rotations+** (translate, stretch, shear, etc)

Many more possibilities. Combine as well!

Q: how to deal with this at **test time**?

- A: transform, test, average

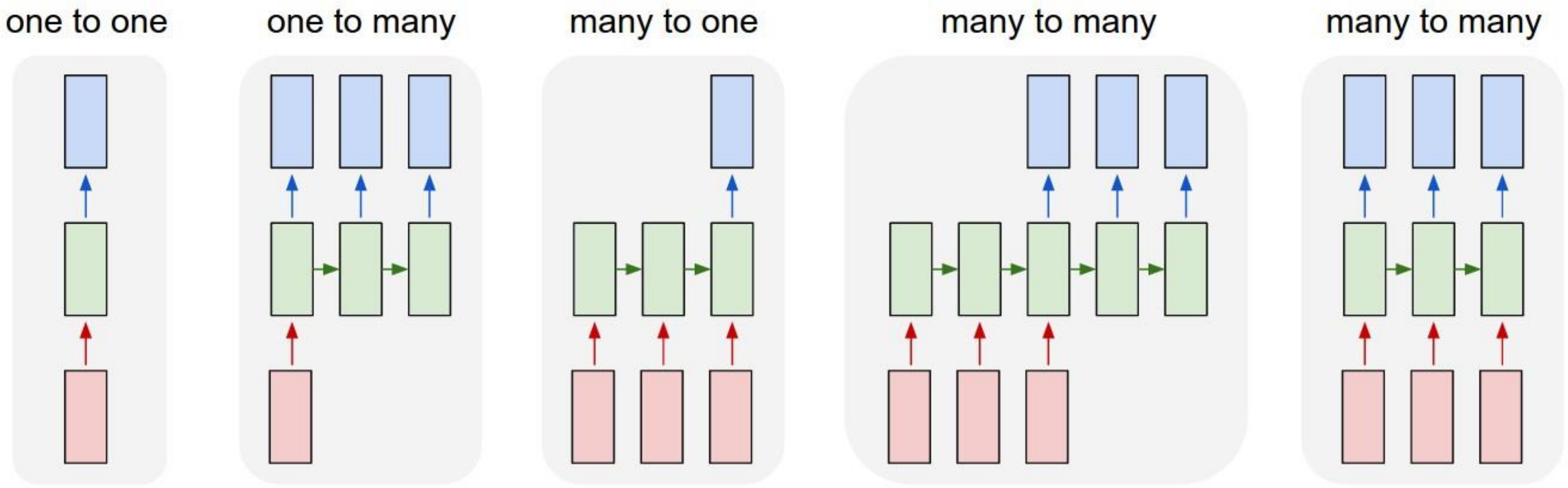


Break & Questions

Outline

- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures (mostly from last time)
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks
- **Graph Models**
  - Data relationships, graph neural networks, graph convolutions

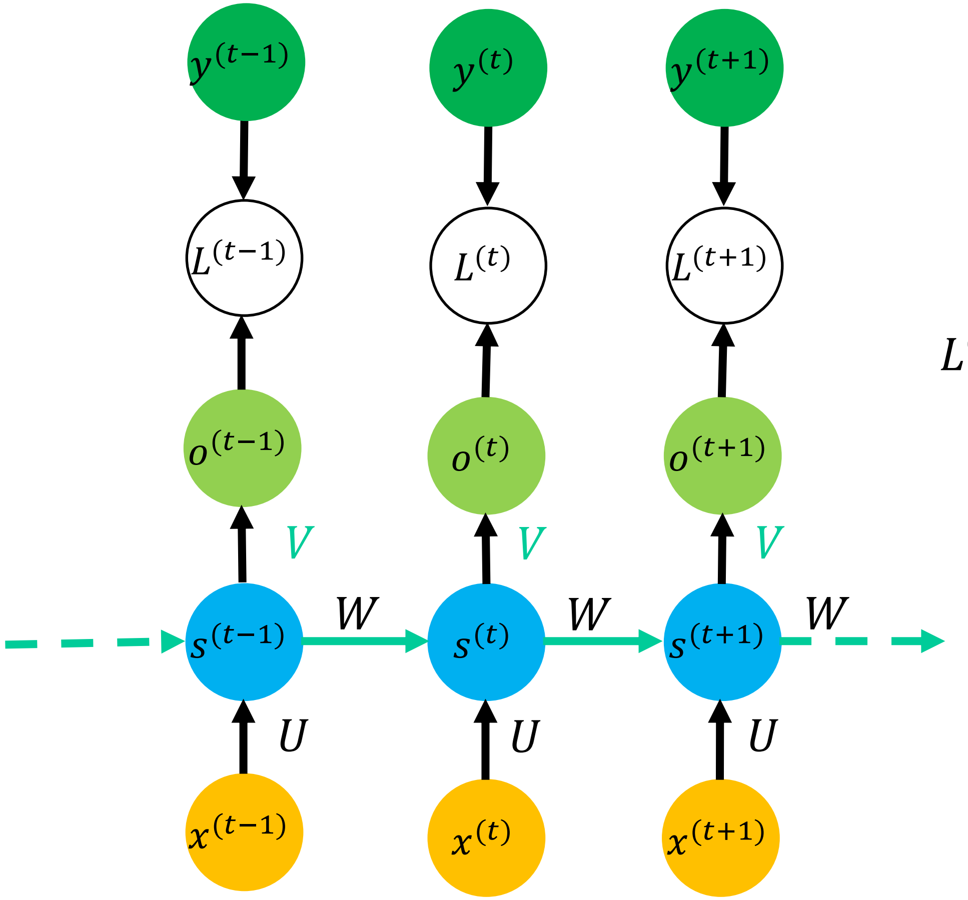
Tasks We Can Handle with NNs?



- Mostly talked about (1) so far
  - Others: need a new kind of model

Neural Networks: Simple RNNs

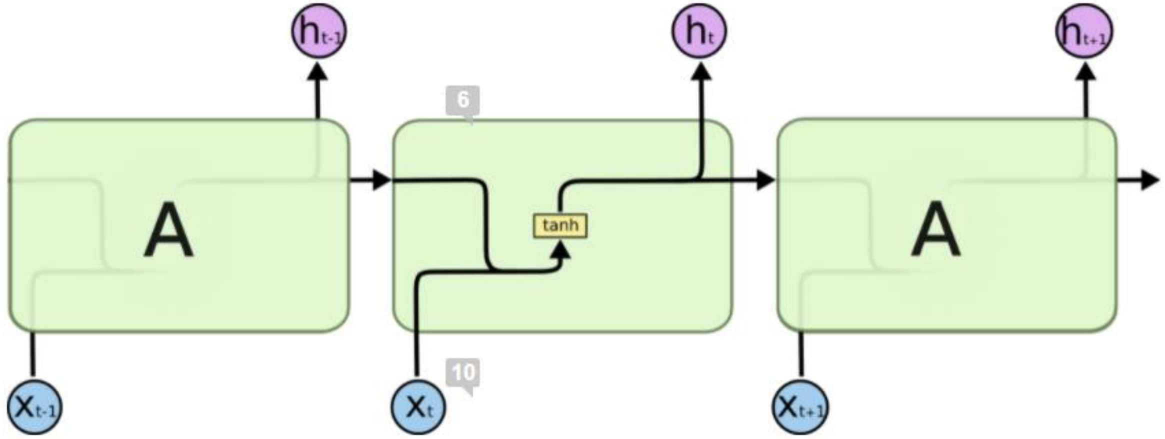
- Classical RNN variant:



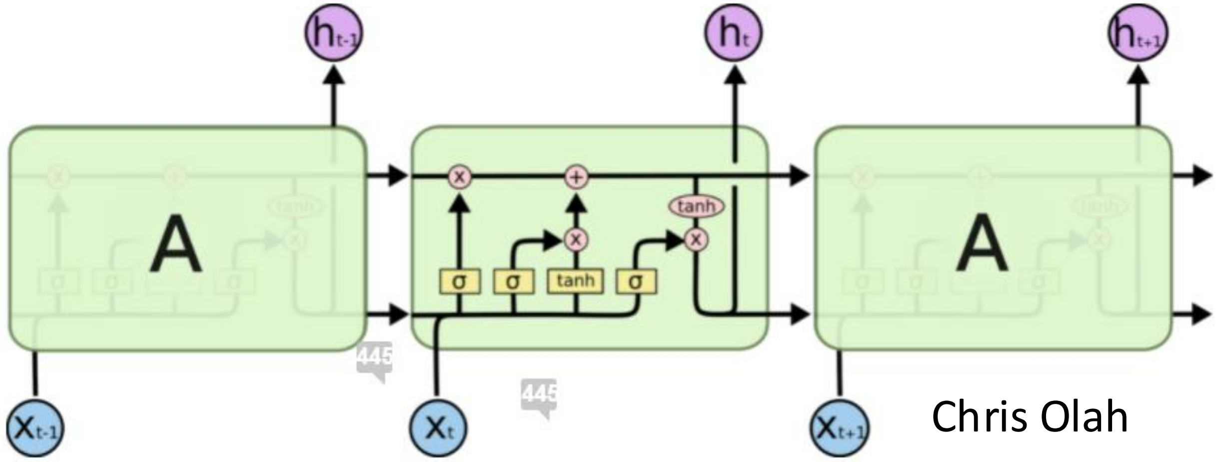
$$a^{(t)} = b + Ws^{(t-1)} + Ux^{(t)}$$
$$s^{(t)} = \tanh\left(a^{(t)}\right)$$
$$o^{(t)} = c + Vs^{(t)}$$
$$\hat{y}^{(t)} = \mathrm{softmax}\left(o^{(t)}\right)$$
$$L^{(t)} = \mathrm{CrossEntropy}(y^{(t)}, \hat{y}^{(t)})$$

Neural Networks: LSTMs

- RNN: can write structure as:



- Long Short-Term Memory: deals with problem. Cell:

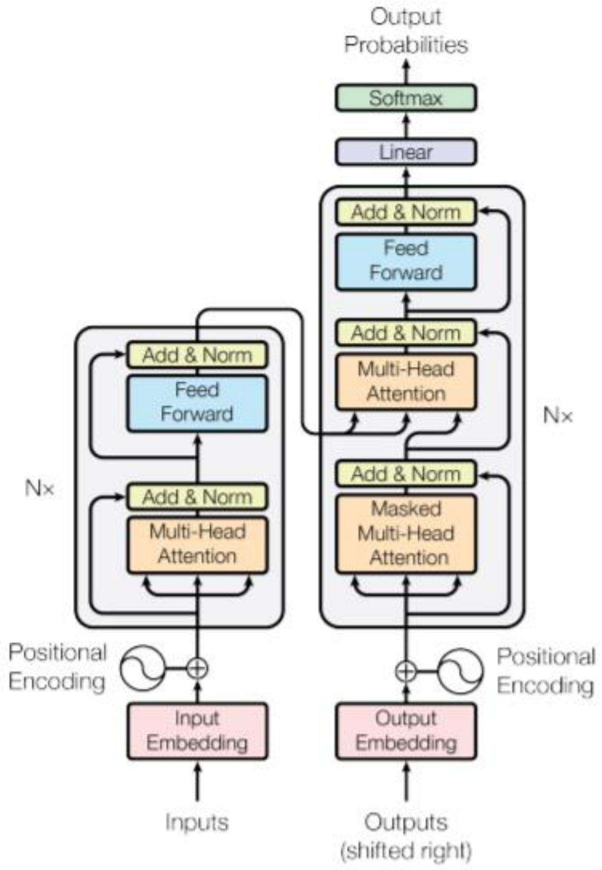


Chris Olah

Neural Networks: Transformers

- Initial goal for an architecture: **encoder-decoder**
  - Get **rid of recurrence**
  - Replace with **self-attention**

- Architecture
  - The famous picture you've seen
  - Centered on self-attention blocks



Vaswani et al. '17

Outline

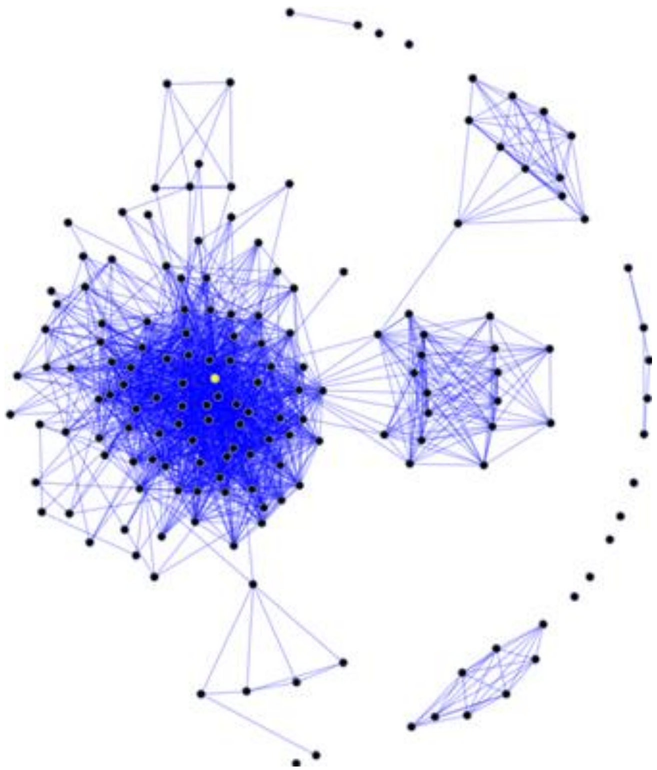
- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures (mostly from last time)
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks
- **Graph Models**
  - Data relationships, graph neural networks, graph convolutions

Break & Questions

Relationships in Data

So far, all of our data consists of points

- Assume all are independent, "unrelated" in a sense $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
- Pretty common to have relationships between points
  - **Social networks**: individuals related by friendship
  - **Biology/chemistry**: bonds between compounds, molecules
  - **Citation networks**: Scientific papers cite each other

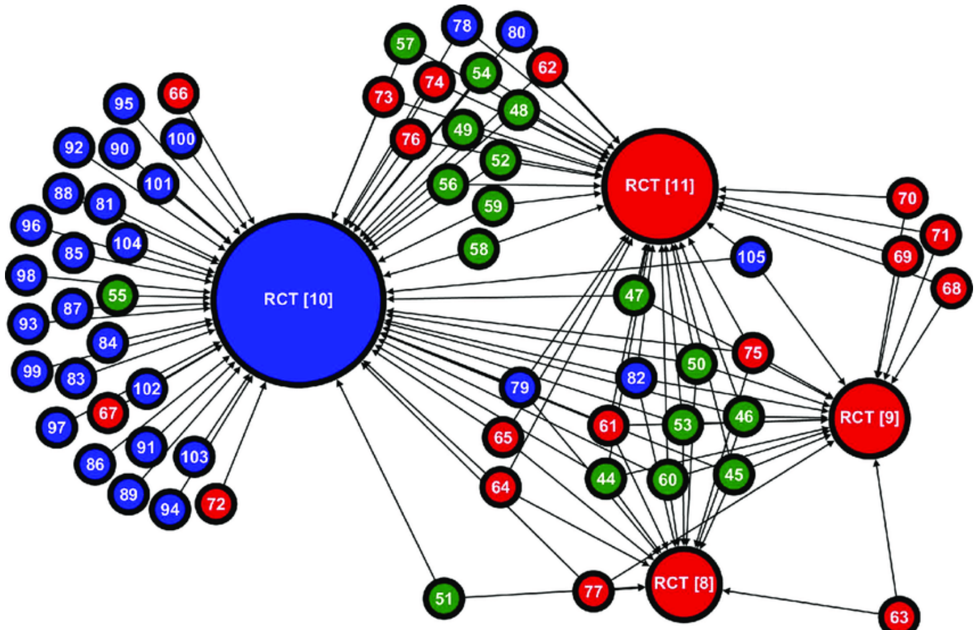


Wiki

Graph Neural Networks: Motivations

- We'll do this via **"graph neural networks"**
- Canonical dataset: **citation networks**.
  - Instances are scientific papers
  - Labels: subfield/genre
  - Graphs: if a paper cites another, there's an edge between them

Note: other features as well (text)



Leng

Graph Neural Networks: Approach

- **Idea**: want to use the graph information in our predictions.
- **Semi-supervised aspect**: don't need all the graph's nodes to be labeled---use network to predict unlabeled nodes.
  - We'll see **much more** of this for foundation models!
- Traditional approach: GNNs keep hidden state at a node

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f(\mathbf{x}_v, \mathbf{x^e}_{(v,u)}, \mathbf{x}_u, \mathbf{h}_u^{(t-1)})$$

Node embedding at time step t

Node features

Edge features

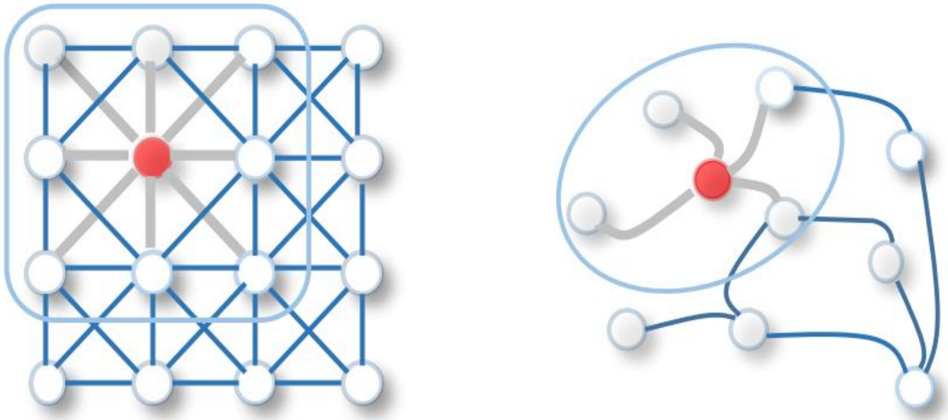
Neighbor embedding at previous step

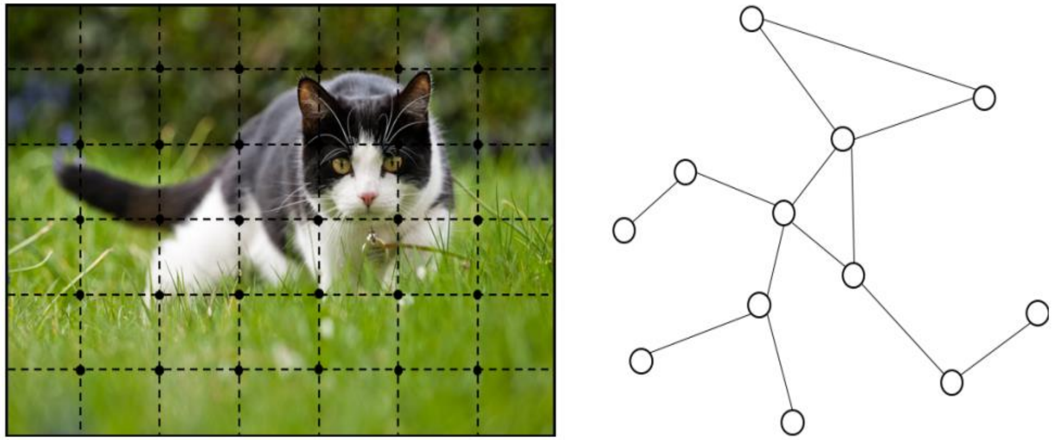
Scarselli et al: "The graph neural network model", 2009.

Improvements: Convolution GNNs

- How do we lift this convolution notion concept to graphs?
- Pixels: arranged as a very regular graph
- Want: allow more general configurations (less regular)

Wu et al, A Comprehensive Survey on Graph Neural Networks





Zhou et al, Graph Neural Networks: A Review of Methods and Applications

Graph Neural Networks (GCNs)

Have: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n), G = (V, E)$

How should our new architecture look?

- Still want layers
  - linear transformation + non-linearity
- Now want to integrate neighbors
- Bottom: graph convolutional network

Parameters

Non-Linearity

Hidden Layer Representation

$$H^{(\ell+1)} = \sigma(H^{(\ell)} W^{(\ell)})$$

$$\big\downarrow$$

$$H^{(\ell+1)} = \sigma(A_G H^{(\ell)} W^{(\ell)})$$

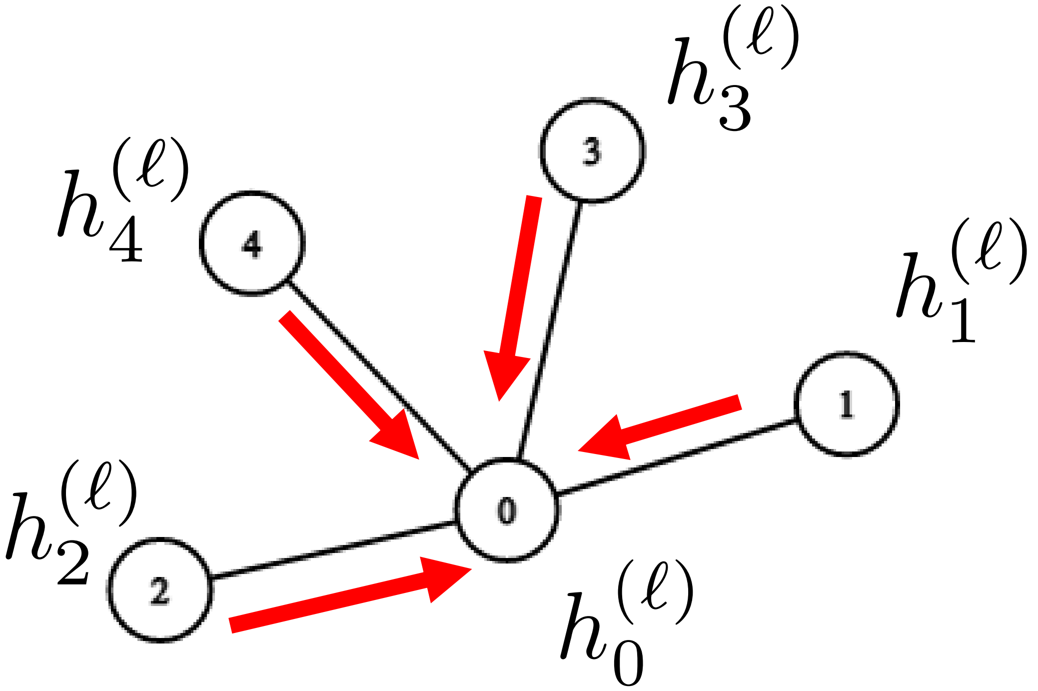
Graph Mixing

Kipf and Welling: “Semi-Supervised Classification with Graph Convolutional Networks”

Graph Convolutional Networks (GCN)

Let's examine the GCN architecture in more detail

- Difference: "graph mixing" component
- At each layer, get representation at each node
- Combine node's representation with neighboring nodes

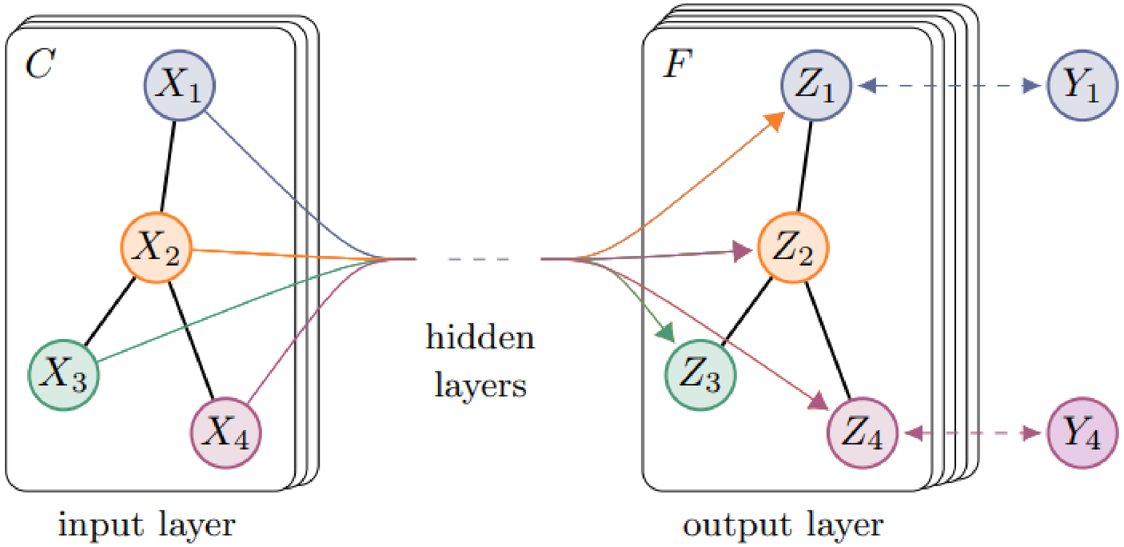


- "**Aggregate**" and "**Update**" rules

Graph Convolutional Networks (GCNs)

- **GCN** two layer network has a very simple form:

$$f(X, A) = \mathrm{softmax}(A\sigma(AXW^{(0)})W^{(1)})$$



Adjacency Matrix

Layer 1 Weights

Layer 2 Weights

Kipf and Welling: "Semi-Supervised Classification with Graph Convolutional Networks"